

Assessing Cognitive Randomness: A Kolmogorov Complexity Approach

Nicolas Gauvrit

LDAR, University Paris VII, Paris, France

Hector Zenil

*LIFL, University of Lille, Lille, France and
Dept. of Computer Science, University of Sheffield, UK*

Jean-Paul Delahaye

LIFL, University of Lille, Lille, France

Abstract

Since human randomness production has been studied and widely used to assess executive functions (especially inhibition), many measures have been suggested to assess the degree to which a sequence is random-like. However, each of them focuses on one feature of randomness, leading authors to have to use multiple measures. Here we describe and advocate for the use of the accepted universal measure for randomness based on algorithmic complexity, by means of a novel previously presented technique using the the definition of algorithmic probability. A re-analysis of the classical Radio Zenith data in the light of the proposed measure and methodology is provided as a study case of an application.

Keywords: algorithmic complexity, subjective randomness, complexity measurement

The production of randomness by humans requires high-level cognitive abilities such as sustained attention and inhibition, and is impaired by poor working memory [1]. Unlike other frontal neuropsychological tests, random generation tasks possess specific features of interest: their demand on executive functions, especially inhibition processes, is high; and more importantly, training does not reduce this demand through automatization. On the con-

trary, generating a random-like sequence requires continuous avoidance of any routine, thus impeding any automatized success.

Random generation tasks have been widely used in the last decades to assess working memory, especially (sustained) inhibitive abilities [2], in normal subjects as well as in patients suffering from a wide variety of pathologies. In normal subjects, random generation varies with personal characteristics or states, such as belief in the paranormal [3] or cultural background [4, 5]. It affords insight into the cognitive effects of aging [6], hemispheric neglect [7], schizophrenia [8], aphasia [9], and Down syndrome [10].

As a rule, random generation tasks involve generating a random-like sequence of digits [11], nouns [6], words [12] or heads-or-tails [13]. Some authors have also offered a choice of more neutral items, such as dots, e.g., in the classical Mittennecker test [14]. Formally however, these cases all amount to producing sequences of bits, that is 0 or 1 digits, since any object can be coded this way. In most studies, the sequence length lies between 5 and 50 items. Measuring the “randomness” of a given short sequence (say one of less than 1000 items) is thus a crucial challenge. Apart from any objective and formal definition of randomness, authors usually use a variety of indices, none of which is sufficient by itself because of the profound limitations they all exhibit. Recently, Schuster, Mittennecker and Papousek [15] provided software calculating the most widely used of such measures applied to the case of the Mittennecker Pointing Test, and provided a comprehensive overview of the usual coefficients of randomness in behavioral and cognitive research.

1. The usual measures of randomness

The usual coefficients used to assess the quality of a pseudo-random production may be classified in three large varieties according to their main focus.

1.1. *Departure from uniformity*

The simplest coefficients – even though they may rely upon sophisticated theories – are based on the mere distribution of outcomes, and are therefore independent of the order of the outcomes. In a word, they amount to the calculation of a distance between the observed distribution and the theoretical flat distribution, just as a chi-square would do.

Information theory [16] has come a long way and is now often used as a ground theory for assessing randomness. Given a finite sequence s of N

symbols repeatedly chosen from a set of n elements, the average symbol information, also called entropy, is given by $H(s) = \sum_i p_i \log_2(p_i)$, where p_i is the relative frequency of item i in the finite sequence. This entropy is maximal when the relative frequencies are all equal, and then amounts to $H_{\max} = \log_2(n)$.

Symbol redundancy (SR; see [15]) is an example of a coefficient arising from information theory. It is defined to be $SR = 1 - H/H_{\max}$ where H is the entropy and n the number of items to be chosen from at each “toss”. SR is no more than a measure of departure from uniformity. A sequence’s SR does not depend on all aspects of the sequence, but only on the relative frequencies of each item comprising it. According to SR, a sequence of 0-1 such as 000001 is weakly random as expected, but 010101, 000111 or 100101 are exactly equivalent to each other, since they all minimize SR to 0. This is the most obvious limitation of SR as a measure of randomness, as well as of any measure relying on the mere distribution of symbols.

1.2. Normality

Beyond values depending on the sole distribution of symbols, one may consider pairs (dyads) or sequences of three (triads) adjacent outcomes. In a truly (infinite) random sequence, any dyad should appear with the same probability. Any distance between the distribution of dyads or triads (and so on) and uniformity may therefore be thought of as a measure of randomness. This is precisely what context redundancy CR_1 and coefficient of constraint CC_1 do [15].

One may also consider dyads of outcomes separated by 1, 2 or k elements in the sequence, which is done, for instance, through CC_k and CR_k coefficients, a generalization of CC_1 and CR_1 . Here we group methods of this kind under the rubric of “normality assessment” for a reason that will soon become clear.

In mathematics, a sequence of digits d_1, \dots, d_n, \dots , with d_n lying within $[0, b]$ – or equivalently the real number written $0, d_1 d_2 d_3 \dots$ in base $(b+1)$ – is said to be normal in base b if the asymptotic distribution of any dyad, triad, or of finite sequences of k consecutive digits, is uniform. Any such series also satisfies what may seem a stronger property: dyads of outcomes separated by a certain fixed number of other outcomes show, in the long run, a uniform distribution. Therefore, a normal sequence will be considered random by any normality assessment method.

Eventually, there will exist sequences produced by simple rules that are normal. The Copeland-Erdős sequence, arising from the concatenation of prime numbers (235711131719...) is an example. The Champernowne sequence [17], a concatenation of all integers from one on (123456789101112131415...), is an even more simple example. There even exist rules to generate absolutely normal sequences, i.e. numbers that are normal in any base b [18].

1.3. Gaps

Another variety of randomness coefficient is worked out using the rank distances between two identical items. For instance, in the sequence 12311, the distances between occurrences of the symbol 1 are 3 and 1. The frequency distribution of repetition distances (gaps) and the median of repetition gap distribution (MdG) are based on the study of the distance between two identical outcomes. They have proved useful in detecting the so-called cycling bias: people tend to repeat an item only after they have used every other available item once [19].

Gap methods are flawed just as normality assessment is: a normal sequence will pass these tests and be considered truly random, even if a naive rule produces it.

1.4. Psychological justifications and limitations

Notwithstanding their potential limitations, the coefficients mentioned above have proved useful in detecting some common biases in random generation. SR-like values capture outcome biases – the over-use of certain symbols [20]. Normality assessment accurately spots alternation biases – the avoidance of using the same symbol twice, e.g., HH or TT – or the inverse *repetition bias*. Context redundancy has also been linked with cognitive flexibility [21], of which it constitutes an estimate. Gaps and related methods would diagnose the cycling bias, a tendency to repeat the same pattern or to exhaust every available symbol before a repetition [19]. For instance, if the available symbols are 1, 2, 3, 4, a subject might choose 1, 3, 4, 2, 1, postponing the second appearance of “1” until after every symbol has occurred once. Repetition avoidance is known to affect outcomes as far as 6 ranks forward [22], a bias that gap methods shed light on.

It is unclear whether these measures happen to capture the basic biases in human random generation, or whether, unfortunately, authors have focused on these biases simply because they have had tools at their disposal for diagnosing them. As we have seen in the previous sections, a normal sequence

such as that suggested by Champernowne, which is highly non-random to the extent that it is generated by a simplistic rule, would meet all random criteria using symbol distribution, normality assessment, and even gap methods.

At this point, we may list three ways in which the usual randomness estimates are flawed: First, they do not capture non-standard bias in randomness, such as the existence of a simple generation rule, providing this rule produces sequences bearing some resemblance to a truly random one, e.g., a normal sequence. Only a few features of a random sequence are captured by these tailored measures. Second, they lack a theoretical basis. Despite being based on formal probabilistic properties, they nevertheless are not subsumed by a theory of randomness. In fact, they neither use nor provide any definition of a random sequence. Third, partly as an upshot of the first two points, several coefficients are needed to sketch an acceptable diagnosis of randomness quality, whereas a single measure would allow the comparison of sequences.

2. Complexity for short sequences

The need for a universal approach that does not focus on specific arbitrary features, as well as a theoretical framework defining randomness, has been expressed by psychologists [23] and addressed outside psychology by the mathematicians Andrei Kolmogorov and Gregory Chaitin. The theory of algorithmic complexity (also known as Kolmogorov-Chaitin complexity) [24] provides a formal definition of a random sequence. In this section, we first provide an overview of this theory, identify its limits, and then suggest an approach for overcoming them, following recent developments in the field [25].

2.1. *The theory of calculability*

When considering the complexity of an object, one may think of said object as simple if it can be described in a few words. One can, for example, describe a string of a million alternating zeros and ones 01010101... as “A million times 01” and say that the string is simple given its short description. However, it is fair to point out that the description of something is highly dependent on the choice of language. The strings a language can compress depend on the language used, since any string (even a random-looking one) can be encoded using a one-word long description by mapping it onto any

word of a suitable language. A language can always be tailor-made to describe any given object by using something that describes said object in a very simple way. Due to these difficulties it wasn't until the arrival on the scene of the theory of computation, and the precise definition of a computing machine by Alan Turing [26], that the theory of algorithmic information found a formal framework on which it could build a definition of complexity and randomness.

Today, the Turing machine model represents the basic framework underlying most, if not all concepts in computer science, including the definition of algorithmic randomness. A Turing machine (henceforth TM) is an abstract model of a digital computer formalizing and defining the idea of mechanical calculation.

A TM consists of a list of rules capable of manipulating a contiguous list of cells (usually pictured as a tape), and an access pointer (an active cell) equipped with a reading head. The TM can be in any one of a finite set of states Q , numbered from 1 to n , with 1 the state at which the machine starts a computation. There is a distinct $n + 1$ state, called the halting state, at which the machine halts. Each tape cell can contain a 0 or a 1 (sometimes there is a special blank symbol filling the tape). Time is discrete and the time instants (steps) are ordered from 0, 1, . . . , with 0 the time at which the machine starts its computation. At any given time, the head is positioned over a particular cell. The head can move right or left, reading the tape. At time 0 the head is situated over a particular cell on the tape called the start cell, and the finite program starts in state 1. At time 0 the content of the tape is called the machine input. With Turing's universal machine ¹ (today named a universal Turing machine, which we will denote simply by UTM) he also proved that programs and data don't have any particular feature that distinguishes them from one another, given that a program can always be the input of another Turing machine, and data can always be embedded in a program. A full description of a Turing machine can be written in a 5-tuples notation as follows: $\{s_i, k_i, s_i + 1, k_i + 1, d\}$, where s_i is the scanned symbol under the head, k_i the state at time t , s_{i+1} the symbol to write at time $t + 1$,

¹Turing's seminal contribution is the demonstration that there are Turing machines capable of simulating any other Turing machine [26]. One does not need specific computers to perform specific tasks; a single programmable computer could perform any conceivable task (we are now capable of running a word processor on the same digital computer on which we play chess games).

k_{i+1} the state to be in at time $t + 1$ and d the head movement either to the right or to the left. A TM can perform the following, which defines one operation: (1) Write an element from $A = \{0, 1\}$. (2) Shift the head one cell left or right. (3) Change to state $k \in Q$.

When the machine is running it executes one such operation at a time (one every step) until it reaches the halting state— if it ever does. At the end of a computation the TM will have produced an output described by the contiguous cells of the tape over which the head passed before halting. A TM may or may not halt, and if it does not halt it is considered to have produced no output. Turing also shows that there is no procedure to determine whether a Turing machine will halt or not [26]. This is set forth as the *undecidability of the halting problem* identified with the common term uncomputability.

2.2. Algorithmic complexity

The basic idea at the root of algorithmic complexity is that a string is random (or complex) if it cannot be produced by a program much shorter in length than itself. The algorithmic complexity $C(s)$ of a bit string s is formally defined as the length in bits of the shortest program that prints out the string running on a UTM U . Formally, $C(s) = \min_p \{|p| : U(p) = s\}$ [27, 28].

One is forced to use a universal Turing machine because one wants a machine capable of printing out any possible string s . However, no general, finite and deterministic procedure exists to calculate $C(s)$, due to Turing's undecidability of the halting problem. $C(s)$ is usually approximated through compression algorithms, such as the Lempel-Ziv algorithm [29]. The length of the compressed string s is actually an upper bound of $C(s)$. However, compression algorithms do not help when strings are short – shorter than, for example, the compression program length in bits.

2.3. The choice of Turing machine matters

The definition of algorithmic complexity clearly seems to depend on the specific UTM U , and one may ask whether there exists a different UTM yielding different values for $C(s)$. The following theorem indicates that the definition of algorithmic complexity makes sense even if measured on different universal Turing machines (or if desired, using different programming languages):

Theorem 1 (invariance). *If U and U' are two UTMs and $C_U(s)$ and $C_{U'}(s)$ the algorithmic complexity of a binary string s using U and U' , there exists a*

constant c that does not depend on s , such that for all binary strings $|C_U(s) - C'_U(s)| < c$.

The proof of this theorem is quite straightforward [30]. The ability of universal machines to efficiently simulate each other implies a comparable degree of robustness. There is a program p_1 for the universal machine U that allows U to simulate U' . One can think of p_1 as a translator in U' for U . Let p_2 be the shortest program producing s according to U' . Then there is a program for U made of p_1 and p_2 and generating s as an output. For strings from a certain length on, this theorem indicates that one will asymptotically approach the same complexity value regardless of the choice of universal Turing machine, provided the complexity will eventually tend toward ∞ .

However, constants can make the calculation of $C(s)$ for *short* strings profoundly dependent on the UTM used [31]. Both the problem of non-computability and the problems posed by short strings may account for the limited number of applications in psychology and other social sciences.

2.4. Algorithmic probability

Solomonoff [32] had the idea of describing the likelihood of a UTM generating a sequence with a randomly generated input program. Formalizing this idea, Levin defined the algorithmic probability of a string s as the probability that a random program (the bits of which are produced with a fair coin flip) would produce s running on a UTM U [33]. Formally Levin's measure is defined as: $m(s) = \sum_{\{p: U(p)=s\}} 1/2^{|p|}$.

For $m(s)$ not to be greater than 1, U has, however, to be what is called a prefix-free Turing machine, that is, a machine that only accepts programs that are not the beginning of any other program (i.e. there exists an "END" sequence finishing any acceptable program). Levin's probability measure induces a distribution over programs producing s , assigning a greater chance that the shortest program is the one actually generating s .

2.5. Algorithmic complexity for short strings

The *coding theorem* connects algorithmic probability to algorithmic complexity [30]:

Theorem 2 (coding theorem). *For any finite string s and a prefix-free universal Turing machine U ,*

$$-\log_2(m(s)) = C_U(s) + O(1)$$

The significance of this theorem lies in the fact that $m(s)$ can be seen as the probability that a universal prefix-free UTM U output s . The coding theorem provides a means to overcome the impossibility of computing $C(s)$ for short strings, given that one can evaluate $C_U(s)$ by calculating $\log_2(m(s))$, using a very natural UTM defined by a lexicographical enumeration of programs (programs enumerated by length). This idea has already been suggested by Delahaye and Zenil [25] as a means of estimating $C(s)$ using $m(s)$ as an alternative to the approximation of $C(s)$ by compression means. To this end, they sampled the space of all possible Turing machines up to a certain size (2 symbols and 4 states). Given the result obtained by Turing mentioned above, enumerating all possible Turing machines and running them one by one is equivalent to using a single universal Turing machine and running it program by program. We denote by $D(n)$ the probability distribution of all 2-symbol n -state Turing machines. In other words, $D(n)$ provides the production frequency of a string s among all 2-symbol n -state TM.

A string is counted in $D(n)$ if it is the output of a Turing machine, i.e. the content of the contiguous cells on the machine tape which the head has gone through before halting.

2.6. Solving the halting problem for small machines

$D(n)$, like $m(s)$ and $C(s)$, is a non-computable function, but one may use the results from a problem popular among computer scientists called the *Busy Beaver problem*.

The Busy Beaver problem is the problem of finding a value $\Sigma(n)$ for the maximum number of 1s a TM with n states can produce before halting, when starting from an empty input. Radò [34] proves that finding $\Sigma(n)$ for any n is impossible given the undecidability of the halting problem, but for $n < 5$ states and 2-symbol TMs, $\Sigma(n)$ is known. These values are arrived at by essentially simulating all different n -state machines and proving that the remaining non-halting machines will never halt (which is possible only because the machines are small and simple).

It is known that $\Sigma(2) = 4$, $\Sigma(3) = 6$ and $\Sigma(4) = 13$ – for 2-symbol TMs – [35, 36].

These values of the Busy Beaver problem allowed Delahaye and Zenil [31] to numerically calculate the complexity of strings in a natural way, using a universal and objective measure of randomness, without having to deal with the problems posed by the impact of additive constants resulting from the

choice of universal Turing machine. The tables produced by Zenil and Delahaye are available online at the following URL: www.algorithmicnature.org

3. An application

The complexity of short sequences based on $D(4)$ may be used to assess their randomness. We will now use it on a classical example, the Radio Zenith experiment data. In 1937, the Radio Zenith foundation carried out an experiment in telepathy: a set of senders picked series of 5 “heads or tails” (or equivalent), which the listeners had to guess. The results seemingly supported the telepathic hypothesis. However, Goodfellow [37] showed that the sequences used in this experiment were not random. For instance, the sequence 00000 or 11111 was used in less than 1% of the trials. The listeners showing the same biases as the senders, the data were shown not to support the paranormal hypothesis in the end.

However, this experiment provides an important dataset for the study of the production of randomness in humans. When subjects try to guess the sequence in the Radio Zenith experiment, we may assume they try not to make it too obvious. Had they tried to be random, their answers would have been uniform – which is not the case. Griffiths and Tenenbaum [38] hypothesized that subjects do not really try to be random, but instead try to maximize the probability that their answers will be random, in a Bayesian way: Let s be the answer (sequence). s may be produced by a rule or machine (event M), or be truly random (event R). What subjects try to maximize is

$$P(R|s) = \frac{P(s|R)P(R)}{P(s|R)P(R) + P(s|M)P(M)}.$$

Our complexity approach to this question provides a means to actually compute such a probability (see Table 1).

A power function gives a good approximation of the link between $P(R|s)$ and the Radio Zenith observed probability of s , $RZ(s)$: $RZ = P(R|s)^{4.37}$; $r = 0.736$; $F = 16.544$; $p < 0.001$. These results support Griffiths and Tenenbaum’s hypothesis, even though (1) many other factors (cultural, cognitive, etc.) may influence the responses, (2) we used a 4-state TM as a model for human cognition (instead of more complex Turing machines) and (3) 5-bit sequences are particularly short. The results also justify the $D(4)$ -measure of complexity/randomness as a good alternative to the multiplication of tailor-made indices focusing on special features of sequences.

Sequence (aggregated)	Zenith Radio data (%)	$P(R s)$ according to $D(4)$ (%)
00000	0.84	34.16
00001	1.39	46.08
00010	3.73	46.77
00011	4.48	57.62
00100	4.99	49.19
00101	14.23	54.21
00110	11.82	53.69
00111	5.66	57.62
01000	3.22	46.77
01001	8.68	50.40
01010	4.34	51.62
01011	5.7	54.21
01100	10.9	53.69
01101	11.66	50.40
01110	6.48	61.13
01111	1.95	46.08

Table 1: For each 5-bit sequence (00000 and 11111 are aggregated, as well as 00110 and 11001, etc.), the table displays the percentage of people who produced the sequence, and the probability that the sequence is random, in a complexity approach using $D(4)$.

Other potential applications of the randomness assessment based on $D(4)$ include (1) the study of classical random-generation biases: Alternation bias, or cycling bias, may well turn out to be a more general complexity bias (over-representation of complex sequences in human pseudo-random sequences) (2) the development of a numerical measure of attention through random generation tasks.

References

References

- [1] J. N. Towse, A. Cheshire, Random number generation and working memory, *European Journal of Cognitive Psychology* 19 (3) (2007) 374–394.
- [2] A. Miyake, N. P. Friedman, M. J. Emerson, A. H. Witzki, A. Howerter, T. Wager, The unity and diversity of executive functions and their contributions to frontal lobe tasks: A latent variable analysis, *Cognitive Psychology* 41 (2000) 49–100.
- [3] P. Brugger, T. Landis, M. Regard, A "sheep-goat effect" in repetition avoidance: Extra-sensory perception as an effect of subjective probability?, *British Journal of Psychology* 81 (1990) 455–468.
- [4] M. Vandewiele, W. D'Hondt, W. Didillon, S. Iwawaki, T. Mwamwendat, Number and color preferences in four countries, *Perceptual and Motor Skills* 63 (2) (1986) 945–946.
- [5] H. Streng, C. B. Lesmana, L. K. Suryani, Random number generation in bilingual balinese and german students: Preliminary findings from an exploratory cross-cultural study, *Perceptual and Motor Skills* 109 (2009) 61–75.
- [6] H. Heuer, M. Janczyk, W. Kunde, Random noun generation in younger and older adults, *The Quarterly Journal of Experimental Psychology* 63 (3) (2010) 465–478.
- [7] T. Loetscher, P. Brugger, Random number generation in neglect patients reveals enhanced response stereotypy, but no neglect in number space, *Neuropsychologia* 47 (2008) 276–279.

- [8] K. Chan, C. Hui, C. Chiu, S. Chan, M. Lam, Random number generation deficit in early schizophrenia, *Perceptual and Motor Skills* 112 (2011) 91–103.
- [9] H. Proios, S. S. Asaridou, P. Brugger, Random number generation in patients with aphasia: A test of executive functions, *Acta Neuropsychologica* 6 (2008) 157–168.
- [10] N. J. Rinehart, J. L. Bradshaw, S. A. Moss, A. V. Brereton, B. J. Tonge, Pseudo-random number generation in children with high-functioning autism and asperger’s disorder, *Autism* 10 (1) (2006) 70–85.
- [11] T. Loetscher, C. Bockisch, P. Brugger, Eye position predicts what number you have in mind, *Current Biology* 20 (6) (2009) 264–265.
- [12] K. I. Taylor, D. P. Salmon, A. U. Monsch, P. Brugger, Semantic and phonemic effects in random word generation: A dissociation between alzheimer’s and huntington’s disease patients, *Journal of the International Neuropsychological Society* 11 (3) (2005) 303–310.
- [13] U. Hahn, P. Warren, Perceptions of randomness: Why three heads are better than four, *Psychological Review* 116 (2) (2009) 454–461.
- [14] E. Mittenecker, Die analyse zuflliger reaktionsfolgen [the analysis of random action sequences], *Zeitschrift fr Experimentelle und Angewandte Psychologie* 5 (1958) 45–60.
- [15] G. Schulter, E. Mittenecker, I. Papousek, A computer program for testing and analyzing random generation behavior in normal and clinical samples: The mittenecker pointing test, *Behavior Research Methods* 42 (2010) 333–341.
- [16] The mathematical theory of communication, University of Illinois Press, 1949.
- [17] D. G. Champernowne, The construction of decimals normal in the scale of ten, *Journal of the London Mathematical Society* 8 (1933) 254–260.
- [18] V. Becher, S. Figueira, An example of a computable absolutely normal number, *Theoretical Computer Science* 270 (2002) 947–958.

- [19] N. Ginsburg, P. Karpiuk, Random generation: Analysis of the responses, *Perceptual and Motor Skills* 79 (1994) 1059–1067.
- [20] R. Nickerson, The production and perception of randomness, *Psychological Review* 109 (2) (2002) 330–357.
- [21] D. Stoffers, H. W. Berendse, J. B. Deijen, E. C. Wolters, Motor perseveration is an early sign of parkinsons disease, *Neurology* 57 (2001) 2111–2113.
- [22] A. Chapanis, Human production of "random" numbers, *Perceptual and Motor Skills* 81 (1995) 1347–1363.
- [23] J. Barbasz, Z. Stettner, M. Wierzhon, K. Piotrowski, A. Barbasz, How to estimate randomness in random sequence generation tasks?, *Polish Psychological Bulletin*, 39 (1), 42-46. 39 (1) (2008) 42–46.
- [24] *An Introduction to Kolmogorov Complexity and Its Applications*, Springer Verlag, 2008.
- [25] *Randomness and Complexity. From Leibniz to Chaitin*, World Scientific Publishing, 2007, Ch. On the Kolmogorov-Chaitin complexity for short sequences, pp. 123–130.
- [26] A. Turing, On computable numbers, with an application to the entscheidungsproblem, *Proceedings of the London Mathematical Society* 2 (42) (1936) 230–265.
- [27] A. Kolmogorov, Three approaches to the quantitative definition of information, *Problems of Information and Transmission* 1 (1) (1965) 1–7.
- [28] G. Chaitin, On the length of programs for computing finite binary sequences, *Journal of the ACM* 13 (4) (1966) 547–569.
- [29] A. Lempel, J. Ziv, On the complexity of finite sequences, *IEEE Transactions in Information Theory* 22 (1) (1976) 75–81.
- [30] *Information and Randomness. An Algorithmic Perspective*, 2nd Edition, Springer-Verlag, 2002.
- [31] J.-P. Delahaye, H. Zenil, Towards a stable definition of kolmogorov-chaitin complexity, retrieved from arXiv:0804.3459v3 [cs.IT] (2008).

- [32] R. Solomonoff, A preliminary report on a general theory of inductive inference, Tech. rep., Zator Co., Cambridge (1960).
- [33] A. Zvonkin, L. Levin, The complexity of finite objects and the algorithmic concepts of information and randomness, UMN = Russian Mathematic Surveys 25 (6) (1970) 83–124.
- [34] T. Radó, On non-computable functions, Bell Sytem Technical Journal 41 (3) (1962) 877–884.
- [35] L. Shen, T. Radó, Computer studies of turing machine problems, Journal of the ACM 12 (2) (1965) 196–212.
- [36] A. Brady, The determination of the value of rado’s noncomputable function $\sigma(k)$ for four-state turing machines, Mathematics of Computation 40 (162) (1983) 647–665.
- [37] L. Goodfellow, A psychological interpretation of the results of the zenith radio experiments in telepathy, Journal of Experimental Psychology 23 (1938) 601–632.
- [38] T. Griffiths, J. Tenenbaum, Randomness and coincidences: Reconciling intuition and probability theory, in: 23rd Annual Conference of the Cognitive Science Society proceedings, 2001, pp. 370–375.